Ramin Tavassoli

Final Project -

Report CS565

Preprocessing:

The training data had information on 960,000 records. For each node, we have up to 20 videos that are recommended. The count for the distinct nodes in the training set amounted to roughly 20 million. To create the graph, I used the NetworkX module recommended by Harshal. I first added all nodes. Then I added edges with weights based on the ranking in the training set. The highest recommended node received an incoming edge of weight 1 etc. For some of the nodes, a few to all of recommendations were missing. For those positions, edges were not added and weights were skipped. For instance if the 15th recommended node was the only one missing from node j, then edges of weight 1 through 14 would be added from node j to the initial 14 nodes in the order, 15th node would be skipped, followed by edges of weights 16 through 20 added to the rest of the nodes in the order. Doing so, the total edge count amounted to roughly 40 million.

Methods:

I implemented a few ideas but the shortest path yielded the highest score.The high level idea was to determine the shortest path length between the source and the target nodes in the test set corresponding to the graph. First all the shortest path lengths were to be computed and stored in a matrix. Then the threshold was to be tuned to optimize score. I shortly encountered two problems. First, the problem was too big and my computer, with python multiprocessing.pool invoked would take 20+ hours to compute all shortest paths. The second disconcerting fact was that 10% of the nodes that appeared in the test set were nonexistent in the training graph.

The first problem was handled by launching an instance in AWS EC2 with 64 CPU cores. This was the first time I had to use a cloud based computing system and it took a while to get it running but when I eventually got it working at 5 AM a few nights back, The results were astonishing. The problem was solved in just under 2 hours. However, I did not launch the program until I solved the second problem described next.

I then launched the program in EC2. As mentioned, in just under 2 hours, I got an array of shortest path lengths associated with each source, target instance in the test set. It was then time to find a threshold of acceptance. The optimal threshold was found at 23 where any path length falling below translated to an edge being present. I then revisited the hypothesis I made about the nonexistent test data nodes. I incrementally allowed more and more edges be present when the instance arose by randomly generating from a uniform distribution. Surprisingly, the best score was achieved when all such cases were assigned an edge meaning all cases in the test data where the node data of one (source or target) was missing were connected.

On the last day, I improved my score to the second place by using Jaccard similarity. I isolated the 20 recommended videos of each source and for each target, determined the videos that recommended the target. Intersecting the two sets and setting a threshold of 1 (if 1 or more videos intersected between sets, an edge was predicted to exist), I achieved a second score set for the test data. I then cross-referenced the two score sets (shortest path length and Jaccard) and for each decision where Jaccard predicted edge presence and the shortest path did not, I overruled the shortest path decision.

Failed/suboptimal Approaches:

Most of my time for this assignment was taken by the idea of graph clustering. Graph clustering is a method involving conductance minimization which algebraically translates into finding the Laplacian and its second eigenvector. Despite storing the 20x20 M Laplacian as a sparse matrix using the Scipy module, finding the second eigenvector was computationally intractable. I ran a fully debugged code for 15+ hours twice overnight and both instances failed to complete. I decided to pursue multiprocessing, but soon discovered the bitter fact that eigenvector computation is not a dividable process, thus rendering the distributed computing approach ineffective. This realization led me to focus on the shortest path length alternative.